First Hit

End of Result Set

L12: Entry 12 of 12 File: JPAB Jun 3, 2004

PUB-NO: JP02004157805A
DOCUMENT-IDENTIFIER: JP 2004157805 A
TITLE: SCHEDULE DEVELOPMENT METHOD, PROGRAM, AND TASK SCHEDULE DEVELOPMENT DEVICE

PUBN-DATE: June 3, 2004

INVENTOR-INFORMATION:

NAME	COUNTRY
TAKEMURA, TSUKASA	

ASSIGNEE-INFORMATION:

NAME	COUNTRY
INTERNATL BUSINESS MACH CORP	

APPL-NO: JP2002323392
APPL-DATE: November 7, 2002

INT-CL (IPC): G06 F 9/44; G06 F 11/28; G06 F 17/60

ABSTRACT:

PROBLEM TO BE SOLVED: To mechanically develop a task schedule for a packaging or testing from design information of a system such as a class view of UML without manpower.

SOLUTION: A task generation part 20 extracts classes constituting the system designed with object-orientation from the design information of the system, and generates a task corresponding to each extracted class. A dependence setting part 30 extracts the dependence between classes from the design information of the system, and sets the dependence between the corresponding tasks based on this dependence. A work load estimation part 40 extracts complicatedness of a certain class from the design information of the system, and estimates a work load required for the packaging of this class based on the complexity. A Gannt chart generation part 50 develops the schedule of the tasks based on these pieces of information.

h e b b g e e e f e e e he h c

(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2004-157805
(P2004-157805A)
(43)公開日　平成16年6月3日(2004.6.3)

(51)Int.Cl.[7]	F I			テーマコード（参考）
G06F 9/44	G06F	9/06	620E	5B042
G06F 11/28	G06F	11/28	340A	5B076
G06F 17/60	G06F	17/60	162A	

審査請求 有　請求項の数 18　OL　（全 15 頁）

(21)出願番号　特願2002-323392(P2002-323392)
(22)出願日　平成14年11月7日(2002.11.7)

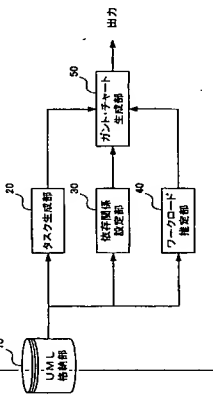
（特許庁注：以下のものは登録商標）
フロッピー

(71)出願人　390009531
インターナショナル・ビジネス・マシーンズ・コーポレーション
INTERNATIONAL BUSINESS MASCHINES CORPORATION
アメリカ合衆国10504　ニューヨーク州　アーモンク　ニュー　オーチャード　ロード
(74)代理人　100086243
弁理士　坂口　博
(74)代理人　100091568
弁理士　市位　嘉宏
(74)代理人　100108501
弁理士　上野　剛史

最終頁に続く

(54)【発明の名称】スケジュール作成方法、プログラム及びタスクスケジュール作成装置

(57)【要約】
【課題】ＵＭＬのクラス図のようなシステムの設計情報から実装やテストのタスクスケジュールを、人手を介さず機械的に作成する。
【解決手段】タスク生成部２０により、オブジェクト指向で設計されたシステムの設計情報からこのシステムを構成するクラスを抽出し、抽出された各クラスに対応するタスクを生成する。また、依存関係設定部３０により、このシステムの設計情報からクラスの間の依存関係を抽出し、この依存関係に基づいて対応するタスクの間の依存関係を設定する。さらに、ワークロード推定部４０により、このシステムの設計情報からクラスの複雑度を抽出し、この複雑度に基づいてこのクラスの実装に要するワークロードを推定する。そして、ガント・チャート生成部５０により、これらの情報に基づいてタスクのスケジュールを作成する。
【選択図】　図２



【特許請求の範囲】
【請求項１】
コンピュータを用いて、オブジェクト指向で設計されたシステムを実現するタスクのスケジュールを作成するスケジュール作成方法であって、
前記システムの設計情報から当該システムを構成するクラスを抽出し、抽出された各クラスに対応するタスクを生成し、生成された当該タスクに関する情報を所定の記憶装置に格納する第１のステップと、
前記システムの設計情報から前記クラスの間の依存関係を抽出し、当該依存関係に基づいて対応する前記タスクの間の依存関係を設定し、設定された当該タスクの間の依存関係に関する情報を所定の記憶装置に格納する第２のステップと、前記所定の記憶装置に格納された前記タスクに関する情報と前記タスクの間の依存関係に関する情報とに基づいて当該タスクのスケジュールを作成する第３のステップと
を含むことを特徴とするスケジュール作成方法。
【請求項２】
前記第２のステップでは、所定のクラス間に、一方のクラスを開発するために他方のクラスが先に開発されている必要があるという関係がある場合に、当該他方のクラスに対応するタスクが終了した後に当該一方のクラスに対応するタスクを開始するように依存関係を設定することを特徴とする請求項１に記載のスケジュール作成方法。
【請求項３】
前記第２のステップでは、所定のクラス間に相互依存の関係がある場合に、相互依存の関係にある全ての当該クラスに対応するタスクが終了できる状態になるのを待って各タスクを終了するように依存関係を設定することを特徴とする請求項１に記載のスケジュール作成方法。
【請求項４】
コンピュータを用いて、オブジェクト指向で設計されたシステムを実現するタスクのスケジュールを作成するスケジュール作成方法であって、
前記システムの設計情報から当該システムを構成するクラスを抽出し、抽出された各クラスに対応するタスクを生成し、生成された当該タスクに関する情報を所定の記憶装置に格納する第１のステップと、
前記システムの設計情報から前記クラスの複雑度を抽出し、当該複雑度に基づいて当該クラスの実装に要するワークロードを推定し、当該ワークロードの推定結果を所定の記憶装置に格納する第２のステップと、
前記所定の記憶装置に格納された前記タスクに関する情報と前記ワークロードの推定結果とに基づいて当該タスクのスケジュールを作成する第３のステップと
を含むことを特徴とするスケジュール作成方法。
【請求項５】
前記第２のステップでは、各クラスにおける過去のワークロードの実績を用い、重回帰分析により当該クラスのワークロードを推定することを特徴とする請求項４に記載のスケジュール作成方法。
【請求項６】
コンピュータを制御して、オブジェクト指向で設計されたシステムを実現するタスクのスケジュールを作成するプログラムであって、
前記システムの設計情報から当該システムを構成するクラスを抽出し、抽出された各クラスに対応するタスクを生成し、生成された当該タスクに関する情報を所定の記憶装置に格納する第１の処理と、
前記システムの設計情報から前記クラスの間の依存関係を抽出し、当該依存関係に基づいて対応する前記タスクの間の依存関係を設定し、設定された当該タスクの間の依存関係に関する情報を所定の記憶装置に格納する第２の処理と、
前記所定の記憶装置に格納された前記タスクに関する情報と前記タスクの間の依存関係に関する情報とに基づいて当該タスクのスケジュールを作成する第３の処理と

を前記コンピュータに実行させることを特徴とするプログラム。
【請求項７】
前記プログラムによる前記第２の処理では、所定のクラス間に、一方のクラスを開発するために他方のクラスが先に開発されている必要があるという関係がある場合に、当該他方のクラスに対応するタスクが終了した後に当該一方のクラスに対応するタスクを開始するように依存関係を設定することを特徴とする請求項６に記載のプログラム。
【請求項８】
前記プログラムによる前記第２の処理では、所定のクラス間に相互依存の関係がある場合に、相互依存の関係にある全ての当該クラスに対応するタスクが終了できる状態になるのを待って各タスクを終了するように依存関係を設定することを特徴とする請求項６に記載のプログラム。
【請求項９】
前記システムの設計情報から前記クラスの複雑度を抽出し、当該複雑度に基づいて当該クラスの実装に要するワークロードを推定し、当該ワークロードの推定結果を所定の記憶装置に格納する第４の処理を、前記コンピュータにさらに実行させ、
前記第３の処理では、前記第４の処理による前記ワークロードの推定結果を加味して前記タスクのスケジュールを作成することを特徴とする請求項６に記載のプログラム。
【請求項１０】
前記プログラムによる前記第４の処理では、各クラスにおける過去のワークロードの実績を用い、重回帰分析により当該クラスのワークロードを推定することを特徴とする請求項９に記載のプログラム。
【請求項１１】
前記第３の処理により作成された前記タスクのスケジュールを記述したガント・チャートを生成して出力する第５の処理を、前記コンピュータにさらに実行させることを特徴とする請求項６に記載のプログラム。
【請求項１２】
前記第１、第２の処理では、ＵＭＬで記述されたクラス図を前記システムの設計情報として用いることを特徴とする請求項６に記載のプログラム。
【請求項１３】
オブジェクト指向で設計されたシステムの設計情報から当該システムを構成するクラスを抽出し、抽出された各クラスに対応するタスクを生成するタスク生成部と、
前記システムの設計情報から前記クラスの間の依存関係を抽出し、当該依存関係に基づいて対応する前記タスクの間の依存関係を設定する依存関係設定部と、
前記システムの設計情報から前記クラスの複雑度を抽出し、当該複雑度に基づいて当該クラスの実装に要するワークロードを推定するワークロード推定部と、
前記タスク生成部により生成された前記タスクに関する情報と前記依存関係設定部にて設定された前記タスクの間の依存関係に関する情報と前記ワークロード推定部による推定結果とに基づいて当該タスクのスケジュールを作成するスケジュール作成部と
を備えることを特徴とするタスクスケジュール作成装置。
【請求項１４】
前記依存関係設定部は、所定のクラス間に、一方のクラスを開発するために他方のクラスが先に開発されている必要があるという関係がある場合に、当該他方のクラスに対応するタスクが終了した後に当該一方のクラスに対応するタスクを開始するように依存関係を設定することを特徴とする請求項１３に記載のタスクスケジュール作成装置。
【請求項１５】
前記依存関係設定部は、所定のクラス間に相互依存の関係がある場合に、相互依存の関係にある全ての当該クラスに対応するタスクが終了できる状態になるのを待って各タスクを終了するように依存関係を設定することを特徴とする請求項１３に記載のタスクスケジュール作成装置。
【請求項１６】

前記ワークロード推定部は、各クラスにおける過去のワークロードの実績を用い、重回帰分析により当該クラスのワークロードを推定することを特徴とする請求項１３に記載のタスクスケジュール作成装置。

【請求項１７】

前記スケジュール作成部は、作成した前記タスクのスケジュールをガント・チャートに記述して出力することを特徴とする請求項１３に記載のタスクスケジュール作成装置。

【請求項１８】

前記システムの設計情報は、ＵＭＬで記述されたクラス図であることを特徴とする請求項１３に記載のタスクスケジュール作成装置。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、オブジェクト指向によるシステム開発において、実装タスクやテスト・タスクのタスクスケジュールを機械的に作成する方法に関する。

【０００２】

【従来の技術】

オブジェクト指向によるシステム設計図を記述する表記方法としてＵＭＬ（Ｕｎｉｆｉｅｄ　Ｍｏｄｅｌｉｎｇ　Ｌａｎｇｕａｇｅ、ＵＭＬは商標）がある（例えば、非特許文献１参照。）。近年、オブジェクト指向のシステム（組織やプロジェクトを含む）開発では、ＵＭＬを用いた分析・設計が盛んに行われるようになっている。

【０００３】

ところで、ＵＭＬを用いて設計されたシステムを実装する場合、クラス間の依存関係に基づいて実装順序を決定する必要がある。例えば、クラス間に継承関係がある場合は、親クラスを子クラスより先に実装しなければならない。また、クラス間に所有関係がある場合は、被所有クラスを所有クラスより先に実装しなければならない。すなわち、被依存クラスを依存クラスよりも先に実装する。当該システムをテストする場合も同様であり、被依存クラスを依存クラスよりも先にテストしなければならない。

【０００４】

従来、システムの実装及びテストを行うためには、人手を介してクラス間の種々の依存関係を考慮し、開発の実行効率の向上を図るために並行して実装できるクラスは並行して実装しながらクラスの実装順序やテスト順序を決定し、タスクのスケジュールを作成していた。

【０００５】

【非特許文献１】

“ＵＭＬ™　Ｒｅｓｏｕｒｃｅ　Ｐａｇｅ”，［ｏｎｌｉｎｅ］，ＯＢＪＥＣＴ　ＭＡＮＡＧＥＭＥＮＴ　ＧＲＯＵＰ，［２００２年１０月７日検索］，インターネット＜ＵＲＬ：ｈｔｔｐ：／／ｗｗｗ．ｏｍｇ．ｏｒｇ／ｕｍｌ／＞

【０００６】

【発明が解決しようとする課題】

上記のように、ＵＭＬを用いて設計されたシステムの実装やテストを行う場合、従来は、人手を介してタスクのスケジュールを作成し、かかる実装やテストを行っていた。

近年のオブジェクト指向技術の普及に伴い、膨大な数のクラスから構成される大規模なシステムが開発されることが多くなってきた。かかる大規模なシステムでは、システムを構成するクラス間の依存関係が複雑になるため、クラスの実装またはテストを行う順序を決定することは多大な手間を要し、誤りが発生する可能性があった。そのため、かかる大規模なシステムにおける実装やテストのタスクのスケジュールを手作業で作成したとしても、クラス間の依存関係の見落としが発生し、クラスの依存関係が十分に反映されない、効率の悪いスケジュールとなる恐れがあった。

【０００７】

そこで本発明は、ＵＭＬのクラス図のようなシステムの設計情報からタスクのスケジュー

ルを、人手を介さず機械的に作成する方法を提案し、これを用いてシステム開発における実装やテストのタスクスケジュールの自動作成を実現することを目的とする。

【0008】

【課題を解決するための手段】

上記の目的を達成する本発明は、オブジェクト指向で設計されたシステムを実現するタスクのスケジュールを作成する、次のようなスケジュール作成方法として実現される。すなわち、このスケジュール作成方法は、ＵＭＬなどで記述されたシステムの設計情報からこのシステムを構成するクラスを抽出し、抽出された各クラスに対応するタスクを生成し、生成されたタスクに関する情報を所定の記憶装置に格納する第１のステップと、このシステムの設計情報からクラスの間の依存関係を抽出し、この依存関係に基づいて対応するタスクの間の依存関係を設定し、設定されたタスクの間の依存関係に関する情報を所定の記憶装置に格納する第２のステップと、以上のステップで記憶装置に格納されたタスクに関する情報とこのタスクの間の依存関係に関する情報とに基づいてこのタスクのスケジュールを作成する第３のステップとを含むことを特徴とする。

【0009】

また、本発明による他のスケジュール作成方法は、システムの設計情報からこのシステムを構成するクラスを抽出し、抽出された各クラスに対応するタスクを生成し、生成されたタスクに関する情報を所定の記憶装置に格納する第１のステップと、このシステムの設計情報からクラスの複雑度を抽出し、この複雑度に基づいてクラスの実装に要するワークロードを推定し、このワークロードの推定結果を所定の記憶装置に格納する第２のステップと、以上のステップで記憶装置に格納されたタスクに関する情報とこのワークロードの推定結果とに基づいてこのタスクのスケジュールを作成する第３のステップとを含むことを特徴とする。

【0010】

また、上記の目的を達成する他の本発明は、次のように構成されたタスクスケジュール作成装置としても実現される。すなわち、このタスクスケジュール作成装置は、オブジェクト指向で設計されたシステムの設計情報からこのシステムを構成するクラスを抽出し、抽出された各クラスに対応するタスクを生成するタスク生成部と、このシステムの設計情報からクラスの間の依存関係を抽出し、この依存関係に基づいて対応するタスクの間の依存関係を設定する依存関係設定部と、このシステムの設計情報からクラスの複雑度を抽出し、この複雑度に基づいてこのクラスの実装に要するワークロードを推定するワークロード推定部と、これらの情報に基づいてタスクのスケジュールを作成するスケジュール作成部とを備えることを特徴とする。

【0011】

上述したスケジュール作成方法及びタスクスケジュール作成装置の機能において、クラス間の依存関係からタスク間の依存関係を設定する場合、より詳しくは、一方のクラスを開発するために他方のクラスが先に開発されている必要があるという関係がある場合に、当該他方のクラスに対応するタスクが終了した後に当該一方のクラスに対応するタスクを開始するように依存関係を設定する。また、所定のクラス間に相互依存の関係がある場合に、相互依存の関係にある全ての当該クラスに対応するタスクが終了できる状態になるのを待って各タスクを終了するように依存関係を設定する。ここで、相互依存の関係には２つのクラス間の相互依存の他に、３つ以上のクラスにおける環状依存の関係を含むものとする。

また、ワークロードの推定を行う場合、より詳細には、各クラスにおける過去のワークロードの実績を用いた重回帰分析などの多変数解析により推定する。ここで、過去のワークロードとは、過去に開発された他のシステムにおける開発時のワークロードや、同一システムにおいて以前に行われたワークロードを含む。

【0012】

さらに好ましくは、上記のようにして作成したスケジュールを記述したガント・チャートを生成して出力することができる。

さらにまた、本発明は、上述したスケジュール作成方法の各ステップに対応する処理をコンピュータに実行させ、あるいはコンピュータを制御して上記のタスクスケジュール作成装置の機能を実現させるプログラムとして実現することができる。このプログラムは、磁気ディスクや光ディスク、半導体メモリ、その他の記録媒体に格納して配布したり、ネットワークを介して配信したりすることにより提供することができる。

【００１３】

【発明の実施の形態】

以下、添付図面に示す実施の形態に基づいて、この発明を詳細に説明する。

図１は、本実施の形態によるタスクスケジュールの自動作成を実現するのに好適なコンピュータ装置のハードウェア構成の例を模式的に示した図である。

図１に示すコンピュータ装置は、演算手段であるＣＰＵ（Ｃｅｎｔｒａｌ ＰｒｏｃｅｓｓｉｎｇＵｎｉｔ：中央処理装置）１０１と、Ｍ／Ｂ（マザーボード）チップセット１０２及びＣＰＵバスを介してＣＰＵ１０１に接続されたメインメモリ１０３と、同じくＭ／Ｂチップセット１０２及びＡＧＰ（Ａｃｃｅｌｅｒａｔｅｄ Ｇｒａｐｈｉｃｓ Ｐｏｒｔ）を介してＣＰＵ１０１に接続されたビデオカード１０４と、ＰＣＩ（Ｐｅｒｉｐｈｅｒａｌ Ｃｏｍｐｏｎｅｎｔ Ｉｎｔｅｒｃｏｎｎｅｃｔ）バスを介してＭ／Ｂチップセット１０２に接続されたハードディスク１０５、ネットワークインターフェイス１０６及びＵＳＢポート１０７と、さらにこのＰＣＩバスからブリッジ回路１０８及びＩＳＡ（Ｉｎｄｕｓｔｒｙ Ｓｔａｎｄａｒｄ Ａｒｃｈｉｔｅｃｔｕｒｅ）バスなどの低速なバスを介してＭ／Ｂチップセット１０２に接続されたフロッピーディスクドライブ１０９及びキーボード／マウス１１０とを備える。

なお、図１は本実施の形態を実現するコンピュータ装置のハードウェア構成を例示するに過ぎず、本実施の形態を適用可能であれば、他の種々の構成を取ることができる。例えば、ビデオカード１０４を設ける代わりに、ビデオメモリのみを搭載し、ＣＰＵ１０１にてイメージデータを処理する構成としても良いし、ＡＴＡ（ＡＴ Ａｔｔａｃｈｍｅｎｔ）などのインターフェイスを介してＣＤ－ＲＯＭ（ＣｏｍｐａｃｔＤｉｓｃ Ｒｅａｄ Ｏｎｌｙ Ｍｅｍｏｒｙ）やＤＶＤ－ＲＯＭ（Ｄｉｇｉｔａｌ Ｖｅｒｓａｔｉｌｅ Ｄｉｓｃ Ｒｅａｄ Ｏｎｌｙ Ｍｅｍｏｒｙ）のドライブを設けても良い。

【００１４】

図２は、本実施の形態によるタスクスケジュール作成装置の機能構成を示す図である。

図２を参照すると、本実施の形態のタスクスケジュール作成装置は、ＵＭＬで記述されたシステム設計図（クラス図）を格納したＵＭＬ格納部１０と、ＵＭＬのクラス図からクラス及びパッケージを抽出してタスクを生成するタスク生成部２０と、ＵＭＬのクラス図におけるクラス間の依存関係を抽出してタスクにおける依存関係を設定する依存関係設定部３０と、ＵＭＬのクラス図における各クラスの複雑度から当該各クラスの実装に要するワークロード（作業量）を推定するワークロード推定部４０と、タスク生成部２０、依存関係設定部３０及びワークロード推定部４０によって得られた情報に基づいてガント・チャートを生成するガント・チャート生成部５０とを備える。

【００１５】

図２に示した構成において、ＵＭＬ格納部１０は、例えば図１に示したコンピュータ装置におけるメインメモリ１０３やハードディスク１０５にて実現される。また、タスク生成部２０、依存関係設定部３０、ワークロード推定部４０及びガント・チャート生成部５０は、プログラム制御されたＣＰＵ１０１にて実現される。

ＣＰＵ１０１を制御してこれらの機能を実現するプログラムは、磁気ディスクや光ディスク、半導体メモリ、その他の記録媒体に格納して配布したり、ネットワークを介して配信したりすることにより提供される。図１に示したコンピュータ装置では、このプログラムがハードディスク１０５に保存（インストール）された後、メインメモリ１０３に読み込まれ展開されて、ＣＰＵ１０１を制御し、上記各種の機能を実現させる。

【００１６】

次に、図２に示したタスクスケジュール作成装置の各機能について説明する。タスク生成

部２０は、ＵＭＬ格納部１０からタスクスケジュールを作成しようとするクラス図を読み出し、ＵＭＬの表記法に従って、当該クラス図を構成するクラス及びクラスを集めたパッケージを抽出する。そして、抽出したクラス及びパッケージに対応するタスクを生成する。本実施の形態ではタスクスケジュールをガント・チャートとして出力するので、抽出されたクラス及びパッケージをガント・チャートにおけるタスクで表現する。タスク名は、例えば抽出されたクラスのクラス名とする。また、親パッケージに含まれるクラス及び子パッケージのタスクは、親パッケージを表すタスクのサブ・タスクとして表現する。生成されたタスクに関する情報は、例えば図１に示したコンピュータ装置におけるメインメモリ１０３やＣＰＵ１０１のキャッシュメモリの所定領域に格納される。

図３は、親パッケージと子であるクラスとの関係を示すＵＭＬのクラス図とそのタスクにおける表現を示す図である。

図３（Ａ）に示すように、パッケージ（Ｐａｃｋａｇｅ）にはクラスＡ、Ｂ、Ｃが含まれている。これをガント・チャートのタスクで表現すると、図３（Ｂ）に示すように、タスクＡ、Ｂ、Ｃがパッケージを表すタスクのサブ・タスクとなる。

【００１７】

依存関係設定部３０は、タスク生成部２０にてＵＭＬ格納部１０から読み出されたＵＭＬのクラス図から、ＵＭＬの表記法に従って、当該クラス図を構成する各クラスの間の依存関係を抽出する。そして、抽出したクラス間の依存関係をタスク生成部２０にて生成されたタスク間の依存関係に変換（設定）する。設定されたタスク間の依存関係に関する情報は、例えば図１に示したコンピュータ装置におけるメインメモリ１０３やＣＰＵ１０１のキャッシュメモリの所定領域に格納される。

クラス間の依存関係として、本実施の形態では、クラス図から各クラスにおける次の関係を抽出する。すなわち、「継承」、「実現」、「所有（集約、コンポジション）」、「依存」、「関連（相互依存）」、「クラスの属性として他のクラスを持つ関係」及び「クラスのメソッド（操作）の引数として他のクラスを使用する関係」である。

【００１８】

上記のクラス間の依存関係についてさらに詳細に説明する。

「継承」では、子クラスが親クラスに依存する。すなわち、子クラスが依存クラス、親クラスが被依存クラスである。この場合、子クラスは親クラスで定義された操作や属性を継承するため、親クラスが先に開発されている必要がある。そこで、ガント・チャートの対応タスクでは、被依存クラスを開発するタスクの終了後に依存クラスを開発するタスクが開始する（「終了－開始」）という依存関係となる。

図４は、「継承」関係にあるクラスとそのタスクにおける表現を示す図である。

図４（Ａ）に示すように、クラスＥ、Ｆが親クラスＤを継承する子クラスである場合、図４（Ｂ）に示すように、子クラスのタスクＥ、Ｆは親クラスのタスクＤに「終了－開始」の関係で依存するようにタスク間の依存関係が設定される。

【００１９】

「実現」では、実装クラスがインターフェイスクラスに依存する。すなわち、実装クラスが依存クラス、インターフェイスクラスが被依存クラスである。この場合、実装クラスはインターフェイスクラスで定義された操作を実装するため、インターフェイスクラスが先に定義されている必要がある。そこで、ガント・チャートの対応タスクでは、「継承」の場合と同様に、被依存クラスを開発するタスクの終了後に依存クラスを開発するタスクが開始するという依存関係となる。

【００２０】

「所有（集約、コンポジション）」では、所有クラスが被所有クラスに依存する。すなわち、所有クラスが依存クラス、被所有クラスが被依存クラスである。この場合、被所有クラスは所有クラスの属性として実装される。すなわち、被所有クラスでは所有クラスへアクセスするための操作を実装する必要がある。この実装のためには被所有クラスが先に開発されている必要がある。そこで、ガント・チャートの対応タスクでは、「継承」や「実現」の場合と同様に、被依存クラスを開発するタスクの終了後に依存クラスを開発するタ

スクが開始するという依存関係となる。
図５は、集約の場合のクラスとそのタスクにおける表現を示す図である。
図５（Ａ）に示すように、クラスＨがクラスＧを所有（集約）する場合、図５（Ｂ）に示すように、タスクＨはタスクＧに「終了―開始」の関係で依存するようにタスク間の依存関係が設定される。
図６は、コンポジションの場合のクラスとそのタスクにおける表現を示す図である。
図６（Ａ）に示すように、クラスＪがクラスＩを所有（コンポジション）する場合、図６（Ｂ）に示すように、タスクＪはタスクＩに「終了―開始」の関係で依存するようにタスク間の依存関係が設定される。
【００２１】
「依存」では、依存クラスが被依存クラスに依存する。この場合、一般的に、依存クラスは被依存クラスを使用するため、依存クラスの実装には被依存クラスが必要である。そこで、ガント・チャートの対応タスクでは、被依存クラスを開発するタスクの終了後に依存クラスを開発するタスクが開始する（「終了―開始」）という依存関係となる。
図７は、「依存」関係にあるクラスとそのタスクにおける表現を示す図である。
図７（Ａ）に示すように、クラスＬがクラスＫに依存する場合、図７（Ｂ）に示すように、タスクＬはタスクＫに「終了―開始」の関係で依存するようにタスク間の依存関係が設定される。
【００２２】
「関連（相互依存）」では、複数のクラスが相互に依存する。そのため、各クラスの実装が全て終了した時点で各クラスをテストできることとなる。そこで、ガント・チャートの対応タスクでは、各タスクがいずれも終了した場合に限り、それぞれのタスクが終了した（「終了―終了」の関係で依存）と考える。
図８は、相互依存する２つのクラスとそのタスクの表現を示す図である。
図８（Ａ）に示すように、クラスＭ、Ｎが相互に依存する場合、図８（Ｂ）に示すように、タスクＭとタスクＮとが「終了―終了」の関係で依存するようにタスク間の依存関係が設定される。すなわち、例えばタスクＭが先に終了できる状態になったとしても、そのまま終了しないで待機し、タスクＮが終了できる状態となってから、タスクＭ、Ｎ共に終了する。
なお、図８には２つのクラスが相互依存する場合について示したが、「関連」の関係における相互依存には、３つ以上のクラスが環状に依存する場合を含む。この場合、環状依存を構成する全てのクラスが終了できる状態となった場合に、各クラスが終了する（「終了」－「終了」の関係で依存）。
【００２３】
「クラスの属性として他のクラスを持つ関係」では、属性を持つクラスが属性となるクラスに依存する。すなわち、属性を持つクラスが依存クラス、属性となるクラスが被依存クラスである。この場合、属性を持つクラスでは属性となるクラスへアクセスするための操作を実装する。この実装のためには属性となるクラスが先に開発されている必要がある。そこで、ガント・チャートの対応タスクでは、被依存クラスを開発するタスクの終了後に依存クラスを開発するタスクが開始する（「終了―開始」）という依存関係となる。
図９は、クラスの属性として他のクラスを持つ関係にあるクラスとそのタスクにおける表現を示す図である。
図９（Ａ）に示すように、クラスＰが属性としてクラスＯを持つ場合、図９（Ｂ）に示すように、タスクＰはタスクＯに「終了―開始」の関係で依存するようにタスク間の依存関係が設定される。
【００２４】
「クラスのメソッド（操作）の引数として他のクラスを使用する関係」では、メソッド（操作）を持つクラスが引数となるクラスに依存する。すなわち、メソッド（操作）を持つクラスが依存クラス、引数となるクラスが被依存クラスである。この場合、メソッド（操作）を定義・実装するためには、引数となるクラスが定義・実装されている必要がある。

そこで、ガント・チャートの対応タスクでは、被依存クラスを開発するタスクの終了後に依存クラスを開発するタスクが開始する（「終了－開始」）という依存関係となる。
図１０は、クラスのメソッド（操作）の引数として他のクラスを使用する関係にあるクラスとそのタスクにおける表現を示す図である。
図１０（Ａ）に示すように、クラスＲのメソッド（操作）の引数としてクラスＱが使用される場合、図１０（Ｂ）に示すように、タスクＲがタスクＱに「終了－開始」の関係で依存するようにタスク間の依存関係が設定される。

【００２５】

ワークロード推定部４０は、タスク生成部２０にてＵＭＬ格納部１０から読み出されたＵＭＬのクラス図から、ＵＭＬの表記法に従って、まず当該クラス図を構成する各クラスの複雑度を調べる。そして、得られたクラスの複雑度に基づいて、当該クラスの実装に要するワークロードを推定する。ワークロードの推定は、タスクスケジュールにおいて時間的なスケールを考慮する上で必要であるが、特に所定のクラスを並行して実装するためには不可欠である。各クラスのワークロードの推定結果は、例えば図１に示したコンピュータ装置におけるメインメモリ１０３やＣＰＵ１０１のキャッシュメモリの所定領域に格納される。
ここで、クラスの複雑度とは、個々のクラスに関連して得られる次の情報とする。すなわち、クラス内の属性（変数等）の数、クラス内の操作（メソッド）の数、クラス内の操作における引数の総数、クラスの持つ関連の数、クラスの持つ集約・コンポジションの数、クラスの持つ依存関係の数、クラスの持つ継承関係の数、クラスの持つ実現関係の数である。なお、これらの情報（パラメータ）は、各クラスのワークロードを推定するために必要な情報であるが、これらに限るものではない。クラスのワークロードを推定するために有効な他の情報があれば、それをクラスの複雑度のパラメータに含めることも可能である。

【００２６】

ワークロードの推定は、重回帰分析などの多変量解析の手法を用いて行うことができる。すなわち、特定のクラスについて実装やテストを行った場合、その際のワークロードの実績を、他のシステム開発や同一システムの改修における同様のクラスの実装やテストに要するワークロードを推定するために用いることができる。この場合、過去に開発された他のシステムにおける開発時のワークロードや、同一システムにおいて以前に行われたワークロード（これらを総称して過去のワークロードと称す）の実績を利用し、次の関係式（重回帰式）により、各クラスの実装に要するワークロードを推定する。
ワークロードＷ＝ｆ（クラスの関連数）＋ｇ（クラスの属性数）＋ｈ（クラスのメソッド数）＋ｉ（Σ（メソッドの引数の数））
上式において、関数ｆ、ｇ、ｈ、ｉは、開発物の技術、経験、開発環境などに依存するので、過去のワークロードにおけるこれらのパラメータに関する実測値に基づいて、多変量解析等の統計的手法を用いて決定される。

【００２７】

ガント・チャート生成部５０は、タスク生成部２０にて生成されたタスク、依存関係設定部３０にて設定されたタスク間の依存関係及びワークロード推定部４０によるワークロードの推定結果から実装タスク及びテスト・タスクのスケジュールを作成するスケジュール作成部である。本実施の形態では、作成したスケジュールをガント・チャートに記述して出力する。図３（Ｂ）、図４（Ｂ）、図５（Ｂ）、図６（Ｂ）、図７（Ｂ）、図８（Ｂ）、図９（Ｂ）及び図１０（Ｂ）は、それぞれのタスクの依存関係を反映させたガント・チャートの例である。
なお、ガント・チャート生成部５０は、後述のように所定のクラス図に対して実装タスクのスケジュールとテスト・タスクのスケジュールとを作成した場合に、各クラスにおいてテスト・タスクの終了後に実装タスクが開始するような依存関係（「終了」－「開始」の関係）を設定する。この依存関係もガント・チャートに反映される。

【００２８】

図１１は、上記のように構成されたタスクスケジュール作成装置による実装タスク及びテスト・タスクのスケジュール作成の動作を説明するフローチャートである。
図１１を参照すると、まず、タスク生成部２０により、ＵＭＬ格納部１０に格納されているＵＭＬのクラス図の中から、スケジュールを作成しようとするクラス図が読み出される（ステップ１１０１）。そして、当該クラス図を構成するクラスが抽出され、対応する実装タスクが生成される（ステップ１１０２）。
次に、依存関係設定部３０により、処理対象であるクラス図を構成する各クラスの間の依存関係が抽出され、これに基づいて各クラスに対応する実装タスクの間の依存関係（開始タイミング及び終了タイミング）が設定される（ステップ１１０３）。
次に、ワークロード推定部４０により、処理対象であるクラス図を構成する各クラスの実装に要するワークロードが推定される（ステップ１１０４）。
そして、ガント・チャート生成部５０により、ステップ１１０２で生成された実装タスク及びステップ１１０３で設定された当該実装タスクの依存関係と、ステップ１１０４で推定された個々のクラスを実装するのに要するワークロードとを反映した、当該実装タスクのスケジュール（ガント・チャート）が作成される（ステップ１１０５）。
【００２９】
次に、同様の手法で、タスク生成部２０によりテスト・タスクが生成され（ステップ１１０６）、依存関係設定部３０によりテスト・タスク間の依存関係が設定され（ステップ１１０７）、ワークロード推定部４０によりワークロードが推定されて（ステップ１１０８）、当該テスト・タスクのスケジュール（ガント・チャート）が作成される（ステップ１１０９）。
そして最後に、各クラスにおいて実装タスクがテスト・タスクに「終了」－「開始」の関係で依存するように、ガント・チャート生成部５０によりタスク間の依存関係が設定され、ガント・チャートに反映される（ステップ１１１０）。
【００３０】
以上の説明では、クラスと実装タスクまたはテスト・タスクとの関係を１対１の対応関係であることを前提として説明したが、１つのクラスから複数のタスクが導出される場合も考えられる。例えば、所定のクラスから、インターフェイスクラスのコーディング・タスク、実装クラスのコーディング・タスク、クラスの単体テスト・タスクが導出される場合である。本実施の形態における依存関係設定部３０は、このような場合の各タスクの依存関係を設定することができる。
図１２は、かかる場合のタスクスケジュール（ガント・チャート）の例を示す図である。図１２において、タスクＡは所定のクラスに対応するテスト・タスク、タスクＢは同一クラスに対応する実装タスクである。また、タスクＡ、Ｂにはそれぞれ、インターフェイスクラスのコーディング・タスク（Ｉｎｔｅｒｆａｃｅ ｃｌａｓｓ ｃｏｄｉｎｇ）、実装クラスのコーディング・タスク（Ｉｍｐｌｅｍｅｎｔａｔｉｏｎ ｃｌａｓｓ ｃｏｄｉｎｇ）、クラスの単体テスト・タスク（Ｕｎｉｔ ｔｅｓｔ）がサブ・タスクとして生成されている。
【００３１】
この場合、依存関係設定部３０は、図示のように、インターフェイスクラスのコーディング・タスクの終了後に実装クラスのコーディング・タスクが開始し、実装クラスのコーディング・タスクの終了後にクラスの単体テスト・タスクが開始するように依存関係を設定する。また、ガント・チャート生成部５０は、Ａクラスのインターフェイスクラスのコーディング・タスクの終了後にＢクラスのインターフェイスクラスのコーディング・タスクが開始し、Ａクラスの実装クラスのコーディング・タスクの終了後にＢクラスの実装クラスのコーディング・タスクが開始し、Ａクラスの単体テスト・タスクの終了後にＢクラスの単体テスト・タスクが開始するように依存関係を設定する。また、図示のように、各タスクは可能な限りにおいて（すなわちタスク間の依存関係を遵守し、かつ並行に実装できる範囲で）並行に実装される。
以上のようにして、本実施の形態では複雑かつ詳細なタスクスケジュールの作成を機械的

に行うことが可能である。

【００３２】

なお、上述した本実施の形態では、タスクスケジュールを作成する対象としてＵＭＬで記述されたシステム設計図（クラス図）を対象として説明したが、本実施の形態は、ＵＭＬ以外にも所定の規則に基づいて記述されたオブジェクト指向のシステムの設計情報に基づいてタスクスケジュールを作成することが可能である。すなわち、当該設計情報の表記法に基づいてシステムを構成するクラスを抽出してタスクを生成し、クラス間の依存関係に基づいてタスクの依存関係を設定し、各クラスのワークロードを推定して、タスクスケジュールを作成する。

【００３３】

また、上述した本実施の形態では、最終的に、作成したタスクスケジュールを記述したガント・チャートを出力したが、ガント・チャートは作成したタスクスケジュールの表現形式に過ぎない。適宜、他の図法等を用いてタスクスケジュールを記述し出力しても良いし、作成されたタスクスケジュールを所定の管理システムにおけるデータとして用い、システム開発におけるスケジュール管理に用いても構わない。

【００３４】

【発明の効果】

以上説明したように、本発明によれば、ＵＭＬのクラス図のようなシステムの設計情報から実装やテストのタスクスケジュールを、人手を介さず機械的に作成することが可能となる。

【図面の簡単な説明】

【図１】本実施の形態によるタスクスケジュールの自動作成を実現するのに好適なコンピュータ装置のハードウェア構成の例を模式的に示した図である。

【図２】本実施の形態によるタスクスケジュール作成装置の機能構成を示す図である。

【図３】親パッケージと子であるクラスとの関係を示すＵＭＬのクラス図とそのタスクにおける表現を示す図である。

【図４】「継承」関係にあるクラスとそのタスクにおける表現を示す図である。

【図５】集約の場合のクラスとそのタスクにおける表現を示す図である。

【図６】コンポジションの場合のクラスとそのタスクにおける表現を示す図である。

【図７】「依存」関係にあるクラスとそのタスクにおける表現を示す図である。

【図８】相互依存する２つのクラスとそのタスクの表現を示す図である。

【図９】クラスの属性として他のクラスを持つ関係にあるクラスとそのタスクにおける表現を示す図である。

【図１０】クラスのメソッド（操作）の引数として他のクラスを使用する関係にあるクラスとそのタスクにおける表現を示す図である。

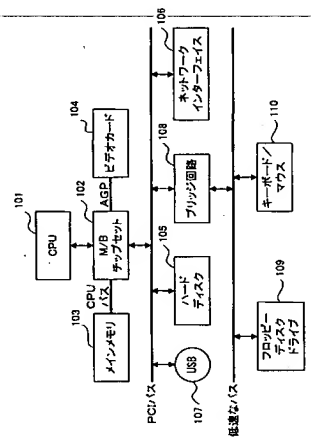
【図１１】本実施の形態における実装タスク及びテスト・タスクのスケジュール作成の動作を説明するフローチャートである。

【図１２】１つのクラスから複数のタスクが導出される場合のタスクスケジュール（ガント・チャート）の例を示す図である。

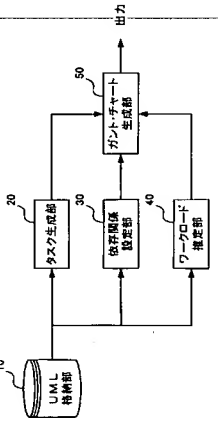
【符号の説明】

１０…ＵＭＬ格納部、２０…タスク生成部、３０…依存関係設定部、４０…ワークロード推定部、５０…ガント・チャート生成部、１０１…ＣＰＵ、１０２…Ｍ／Ｂチップセット、１０３…メインメモリ、１０５…ハードディスク

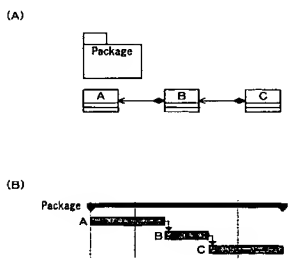
【図１】


101
CPU
102
AGP
N/B チップセット
104
ビデオカード
CPU バス
103
メインメモリ
106
ネットワーク インターフェイス
108
ブリッジ回路
105
ハード ディスク
107
USB
PCIバス
110
キーボード/ マウス
109
フロッピー ディスク ドライブ
低速なバス

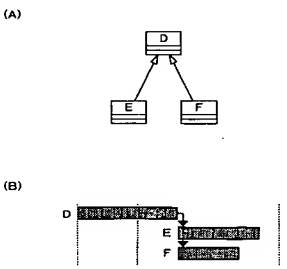
【図２】


出力
50
ガントチャート 生成部
20
タスク生成部
30
依存関係 設定部
40
ワークロード 推定部
10
UML 格納部

【図３】


(A)
Package
A
B
C
(B)
Package
A
B
C

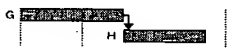
【図４】


(A)
D
E
F
(B)
D
E
F

【図５】

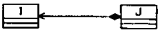
(A)

(B)



【図６】

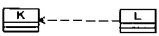
(A)



(B)

【図７】

(A)



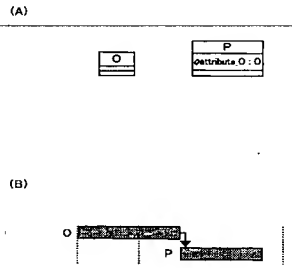
(B)

【図８】

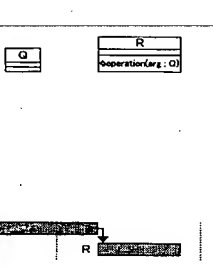
(A)

(B)

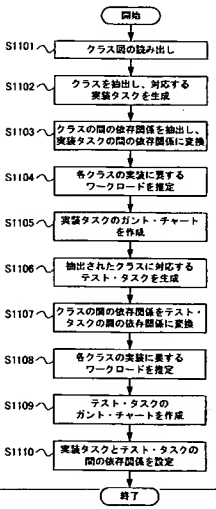
【図９】



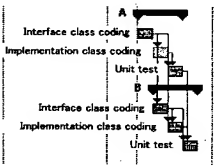
【図１０】



【図１１】



【図１２】



フロントページの続き

(74)復代理人 100104880
弁理士 古部 次郎
(74)復代理人 100118201
弁理士 千田 武
(72)発明者 竹村 司
神奈川県大和市下鶴間１６２３番地１４ 日本アイ・ビー・エム株式会社 大和事業所内

Ｆターム(参考) 5B042 GA08 HH11
5B076 DE03